

# Fine-Tuning a MAP Error Correction Algorithm for Five-Key Chording Keyboards

Adrian Tarniceriu, Bixio Rimoldi, and Pierre Dillenbourg

**Abstract** Different typing devices lead to different typing error patterns. In addition, different persons using the same device have different error patterns. Considering this, we propose and evaluate a spelling algorithm specifically designed for a five-key chording keyboard. It uses the maximum a posteriori probability rule, the probabilities that one character is typed for another, named confusion probabilities, and a dictionary model. Our study shows that the proposed algorithm reduces the substitution error rate from 7.60% to 1.25%. In comparison, MsWord and iSpell reduce the substitution error rates to 3.12% and 3.94%, respectively. The error rate can be further reduced to 1.15% by using individual confusion matrices for each user.

**Key words:** error correction; chording keyboard; maximum a posteriori probability; confusion matrix

## 1 Introduction

With the technological progress, computing devices have become smaller, portable, or mobile. Due to size limitations, the classic QWERTY keyboard became a sub-optimal solution, and was replaced by other methods such as  $4 \times 3$  multi-tap keypads, mini-QWERTY, or touchscreen keyboards. Though easy to use and efficient, these interfaces are not suitable for certain situations. For example, it is difficult to type a text message while walking. Even so, more than 40% of people do it [15],

---

Adrian Tarniceriu

Ecole Polytechnique Fédérale de Lausanne, e-mail: [adrian.tarniceriu@epfl.ch](mailto:adrian.tarniceriu@epfl.ch)

Bixio Rimoldi

Ecole Polytechnique Fédérale de Lausanne e-mail: [bixio.rimoldi@epfl.ch](mailto:bixio.rimoldi@epfl.ch)

Pierre Dillenbourg

Ecole Polytechnique Fédérale de Lausanne e-mail: [pierre.dillenbourg@epfl.ch](mailto:pierre.dillenbourg@epfl.ch)

which is potentially dangerous as the visual attention is committed to typing and not to the surrounding environment.

Chording keyboards [9] represent a solution for the aforementioned situations. These keyboards allow users to generate a character by simultaneously pressing a combination of keys, similarly to playing a note on a musical instrument. Compared to other devices (such as desktop keyboards, mobile-phone keypads, or touchscreens), they require a smaller number of keys. With five keys, there are 31 combinations in which at least one key is pressed, enough for the 26 letters of the English alphabet and five other characters. If the keys are adequately placed in a position that fits naturally under the fingertips, then we can type with only one hand and without looking at the input device. Therefore, we will be able to use a mobile device even during activities for which vision is partially or entirely committed, like walking in crowded spaces, jogging, or riding a bike.

Besides typing fast and with low error rates, it is important to be able to use a text-entry method without too much training. Previous studies [16, 17] showed that people can learn to type with a five-key chording keyboard in less than 45 minutes. After 350 minutes of practice, the average typing rate was around 20 words per minute (wpm) with a maximum of 31.7 wpm, comparable to iPhone, Twiddler [8] or handwriting. The typing error rate at the end of the study was 2.69%. Being able to automatically correct these mistakes will probably increase the keyboard's ease-of-use and typing speed, because users will not have to stop typing in order to correct errors. In addition, being focused on another activity while typing will probably lead to more errors, so efficient error correction becomes even more important in these situations.

This paper continues our work [18] on an error correction mechanism for chording keyboards. The correction mechanism is based on the maximum a posteriori probability principle (MAP) [5] and for each typed word, it provides a list of possible candidates and chooses the one that is the most likely. Moreover, it takes into consideration the particularities of the text input device. This is motivated by the fact that different devices lead to different error patterns, and knowledge about these patterns can be used to improve the error correction methods. Besides presenting the correction algorithm, we also analyze the factors that influence the algorithm's efficiency.

The paper is organized as follows. Section 2 presents a brief overview of existing text error correction mechanisms. In Sects. 3 and 4, we describe the proposed error correction algorithm and the data set used for evaluation. Section 5 outlines the error correction results. In Sect. 6, we conclude the paper and discuss future research directions.

## 2 Related Work

A detailed overview of commonly used correction techniques is presented by Kukich in [7]. Research in spelling error detection and correction is grouped into three main categories:

1. Non-word error detection: Groups of  $n$  letters ( $n$ -grams) are examined and looked up in a table of statistics. The strings that contain non-existing or highly infrequent  $n$ -grams are considered errors.
2. Isolated word error correction: Each word is treated individually and considered either correct or incorrect. In the latter case, a list of possible candidates is proposed. These candidates can be provided using several techniques such as minimum edit distance [19], similarity key techniques [10], rule-based techniques [20],  $n$ -gram techniques [11], probabilistic techniques [4], or neural net techniques [12].

Most isolated word error correction methods do not correct errors when the wrongly typed word is contained in the dictionary. For example, if *farm* is typed instead of *form*, no error will be detected. Moreover, these methods cannot detect the use of wrongly inflected words (for example, *they is* instead of *they are*).

3. Context dependent error correction: These methods try to overcome the drawbacks of analyzing each word individually by also considering the context. Errors can be detected by parsing the text and identifying incorrect part-of-speech or part-of-sentence  $n$ -grams [13]. Or, if enough memory and processing power are available, tables of word  $n$ -grams can be used. Other approaches consider grammatical and inflectional rules, semantical context, and can also identify stylistic errors.

Most of the methods presented above can be applied to any typed text, regardless of the input device. As various input techniques become more and more popular, the classic correction techniques have been improved to consider both the text and the device particularities. Goodman et al. [3] present an algorithm for soft keyboards that combines a language model and the probabilities that the user hits a key outside the boundaries of the desired key. Kristensson and Zhai [6] propose an error correction technique for stylus typing using geometric pattern matching. The T9 text input method for mobile phones can also be included here, as it considers the correspondence between keys and characters to predict words. A strategy that can be applied to chording text input is presented by Sandnes and Huang in [14].

## 3 Algorithm

Traditionally, text error detection and correction focus on character-level errors, which can be classified into three categories: *deletions*, when a character is omitted; *insertions*, when an additional character is inserted; *substitutions*, when a character is substituted by another character.

The algorithm that we propose is designed only for substitution errors and focuses on individual words, without considering any contextual information. It is based on the maximum a posteriori probability principle, taking into account a dictionary model and the probabilities that one character is typed for another.

For a typed word  $y$ , the MAP algorithm will find the string  $\hat{x}$ , which is the most likely in the sense of maximizing the posterior probability  $p(x|y)$  over all  $x \in S$ . The set  $S$  contains all the possible candidate strings. Then,

$$\hat{x} = \arg \max_{x \in S} p(x|y) \quad (1)$$

$$= \arg \max_{x \in S} p(y|x)p(x), \quad (2)$$

where (2) follows from Bayes' rule and takes into account that we maximize with respect to  $x$ .

As we focus on substitutions, we can limit the candidate set to words with the same length as the typed word. Considering this and assuming that error events are independent, we can write

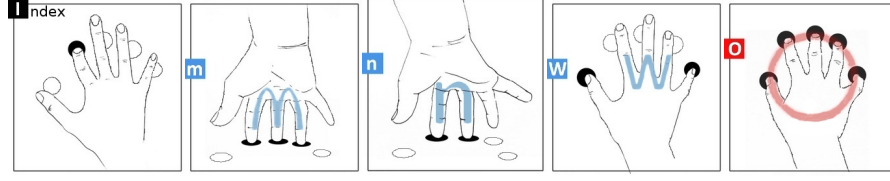
$$p(y|x) = \prod_{i=1}^{i=N} p(y_i|x_i), \quad (3)$$

where  $y_i$  is the  $i$ th letter of the typed word,  $x_i$  is the intended letter, and  $N$  is the word length.  $p(y_i|x_i)$ , named confusion probability, is the probability that the character  $y_i$  is typed in lieu of  $x_i$ . The prior probability,  $p(x)$ , is given by the frequencies of the dictionary entries in English language. For example, given the typed word  $y = oat$  and the candidate  $x = bat$ , we need to compute

$$F(oat|bat) = p(o|b)p(a|a)p(t|t)p(bat). \quad (4)$$

The set  $S$  contains dictionary words with the same length as the typed word. To increase speed, we can use the fact that only a certain fraction of the substitutions occur with non-negligible probability. To describe how this is done, it is useful to represent each character by a five-bit codeword. We choose the first digit to represent the key under the thumb, the second to represent the key under the index, etc. The value of a position is 1 if the corresponding key is pressed and 0 otherwise. So, for instance, 10111, corresponding to the letter  $b$ , means that all fingers except the index are pressing the keys (five examples of mappings between key combinations and characters are shown in Fig. 1). In this way, two words can be compared also from a bit distance point of view, as shown in Table 1. Our tests have shown that in 98.5% of the cases, the wrongly typed word differs from the intended one by at most five bits. Hence, for each typed word, we limit the set  $S$  to words that differ by at most five bits. Compared to using the edit distance, this method provides less candidates, thus increasing the speed of the algorithm.

In our study, we used the British National Corpus, containing approximately 100 million words [1]. The used dictionary was obtained from this corpus by choosing



**Fig. 1** Examples of letter mappings. “i” is given by the initial of the finger pressing the key (index). “m” and “n” are given by the shape of the fingers pressing the keys. “w” is given by the shape of the fingers not pressing the keys. For “o”, we imagine five dots spread around a circle, and we obtain it by pressing all the keys.

**Table 1** Possible candidates for the typed word *oat*

Possible candidate	Binary form	Bit distance
<i>oat</i>	11111 00110 10000	0
<i>bat</i>	11110 00110 10000	1
<i>rat</i>	00010 00110 10000	4

all the items occurring more than five times. It contains 100944 entries, which include inflected forms such as declensions and conjugations. The prior probabilities were given by the word frequency in the corpus and the confusion probabilities were estimated experimentally.

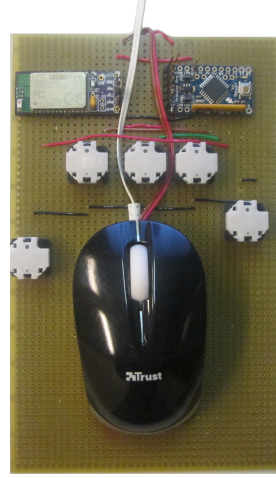
## 4 Evaluation Data

In order to gather enough data to evaluate the proposed algorithm, we asked 10 students from our university to type using a chording keyboard prototype. The prototype has the keys placed around a computer mouse and is presented in Fig. 2. We designed the prototype in this way because we wanted the subjects to see a practical application of a chording device: allowing typing and screen navigation at the same time, with only one hand. The buttons are placed so that they can be easily operated while holding the mouse with the palm. The keyboard is designed using an Arduino Pro Mini microcontroller board and communicates with the computer by Bluetooth.

The total amount of data gathered during the experiment consists of 40 345 words, out of which 4052 (10.17%) contain errors. Of these, 3065 (75.64%) are substitution errors. The remaining 987 errors occurred when people did not type a letter (e.g. *hous* instead of *house*), typed an extra letter (*housee* instead of *house*), the space between words was missing (*thehouse* instead of *the house*), or when whole words were missing, added, or the topic of the sentence changed.

The total number of typed characters is 219 308, from which 5889 are errors. We used these characters to determine the confusion matrix, which is a square matrix with rows and columns labeled with all the characters that can be typed. The value at position  $ij$  shows the frequency of character  $j$  being typed when  $i$  was intended. The

**Fig. 2** Chording keyboard prototype used during the typing study



values are given as percentages from the total number of occurrences for character  $i$  and represent the confusion probabilities used by the algorithm.

## 5 Results

The error-correction algorithm was implemented in MATLAB. To evaluate it, we checked the substitution error rates (the number of words containing substitution errors divided by the total number of typed words) before and after applying the algorithm. In the past [18], we compared the results to MsWord and iSpell, to have a reference for the proposed correction method. As this is important for showing the algorithm's error correction ability, we repeat this comparison in Sect. 5.1, with an improvement in the correction method. In Sects. 5.2, 5.3, and 5.4, we show how different dictionaries and confusion matrices affect the correction efficiency, the distribution of errors for different word lengths, and how to improve the correction mechanism using word bigrams.

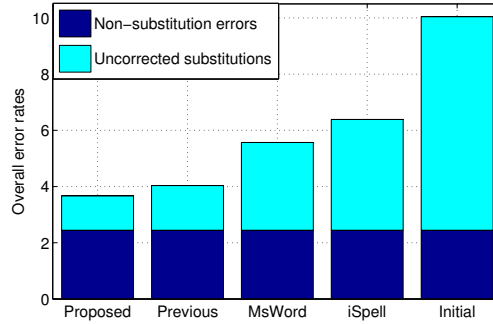
### 5.1 Correction Rates

In our previous work [18], we compared the correction results to MsWord and iSpell. The substitution error rate was decreased from 7.60% to 1.59%, which is considerably better than the two references (3.12% for MsWord and 3.94% for iSpell). During that study, the confusion probabilities,  $p(y_i|x_i)$ , were lower bounded to a fixed value (0.2%), to ensure that any transition between letters is considered. This bound is useful if we have only a small amount of training data, but after experi-

menting with larger amounts of data, we noticed that the transitions between certain characters are accurately described by values lower than 0.2%. Considering this, in the following we set the lower bound to be equal to the lowest non-zero confusion probability.

This change in the lower bound reduces the substitution error rate from 1.59% to 1.25%. The correction results (for the proposed algorithm, for the previous work, for MsWord and for iSpell) are shown in Fig. 3, with the substitution error rates depicted by the light bars. The non-substitution errors (dark bars) are not affected by the correction algorithm.

One should not forget that the dictionaries used by the three methods are not the same, and this can affect the results. Moreover, our algorithm is specifically designed for a five key chording keyboard, while MsWord and iSpell can be applied to any text input device with the same results.



**Fig. 3** Overall error rates for the proposed algorithm, previous work, MsWord and iSpell

The MAP algorithm minimizes the error probability by choosing the candidate with the highest posterior probability, which depends on the prior probability and on the confusion matrix. Therefore, by finding better estimates for the prior probabilities and for the confusion matrix, we will be able to further reduce the error rate.

## 5.2 Dictionary Effect

Besides the error correction rate, we are also interested in the speed of the algorithm. A larger dictionary will most likely reduce the error rates, but the algorithm running time will increase. From a practical point of view, this means slower response times and higher power consumption. Therefore, it is important to find a trade-off between dictionary size and acceptable error rates.

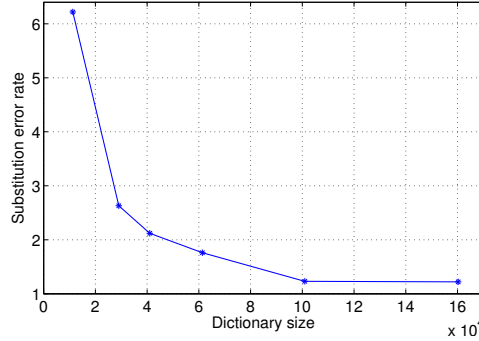
We evaluated the error rate vs. dictionary size for six different dictionaries, denoted as  $D_2$ ,  $D_5$ ,  $D_{20}$ ,  $D_{50}$ ,  $D_{100}$  and  $D_{500}$ , respectively. The subscript of the letter

$D$  shows the minimum number of appearances of every dictionary entry in the used corpus. All of them contain inflected words and their sizes are given in Table 2.

**Table 2** Dictionary sizes

	$D_2$	$D_5$	$D_{20}$	$D_{50}$	$D_{100}$	$D_{500}$
Size	160 250	100 944	61 364	41 028	29 066	11 288

The results are shown in Fig. 4. As expected, the number of errors decreases as the dictionary size increases, but the relation is not linear. Increasing the size above a certain value has only a marginal effect on the error rate, which may not compensate for the increases in time, processing power, and memory requirements. For example, increasing the dictionary size from 100 944 to 160 250 only reduces the error rate from 1.25% to 1.23%, while increasing the dictionary size from 61 364 to 100 944 reduces the error rate from 1.76% to 1.25%. Finding the optimal dictionary size depends on the desired correction rates and available resources. From a fundamental point of view, reducing the dictionary size by excluding words with low frequency in the corpus is equivalent to setting their prior probabilities to zero. Using less accurate priors can only increase the error probability.



**Fig. 4** Substitution error rates for different dictionary sizes

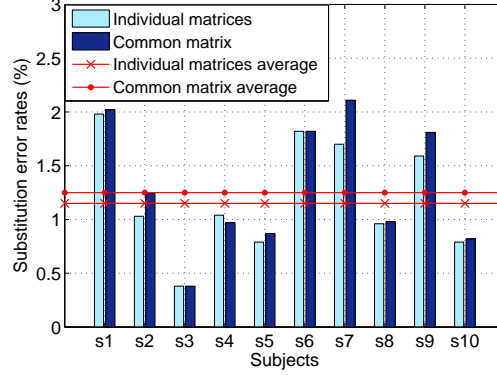
### 5.3 Confusion Matrix Effect

As already mentioned, the performance of the algorithm depends on the accuracy of the confusion matrix. Given the importance of this matrix, and having in mind that it was determined using the data from all the 10 participants, one might expect that using personalized confusion matrices would lead to better results. To test this, we constructed individual confusion matrices from the text typed by each user. Then, we corrected each user's errors using these matrices. The results were ob-



tained using 10-fold cross-validation and are given in Fig. 5, where each group of bars represents the substitution error rates for each user, with the individual and with the common matrix, respectively. The horizontal lines are the substitution error rates for the whole typed text, with individual and with common matrices, respectively.

**Fig. 5** Post-processing substitution error rates with individual and with common confusion matrices



The use of individual confusion matrices decreased the substitution error rate from 1.25% to 1.15% (this represents a relative reduction of 8%). For 7 of the 10 participants in our typing study the error rates are lower than when using the common matrix. For two participants the error rates are the same, and for only one participant the error rates are higher.

One may also expect that, as people gain typing experience, they will make different types of mistakes. Hence, we built the confusion matrices for each session and used them to correct the errors for that specific session (again, by using 10-fold cross-validation). However, in this case we did not notice any improvement in the error rates.

#### 5.4 Word Length Effect

As shown in the second column of Table 3, the average number of candidates considered by the algorithm depends on the typed word's length, being higher for shorter words. This is explained by remembering that each letter can be written as a sequence of five bits, and that for shorter strings it is more likely to obtain a string with non-zero prior probability by modifying a small number of bits. For instance, for length one strings, i.e. single characters, if we look at candidates that are at a distance of five bits or less, we find all the 26 characters of the alphabet.

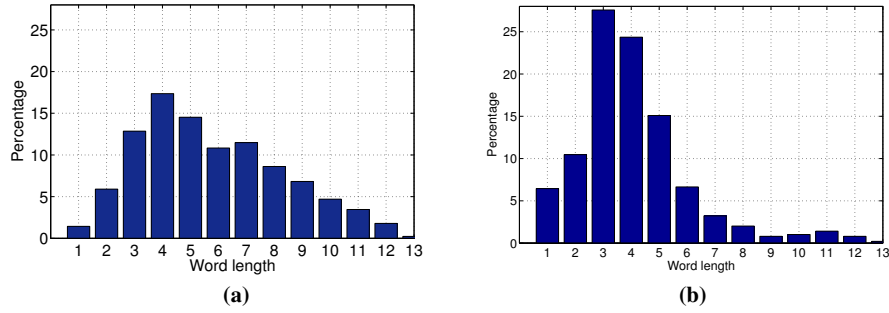
Because there are more candidates for shorter words, the differences between the posterior probabilities of the candidate words are smaller, implying that the error probability is higher. This is confirmed by the third column of Table 3, showing the ratio between the number of substitution errors that remained after applying the

algorithm, and the number of initial substitution errors, named uncorrected ratio. This ratio is higher for shorter words, meaning that the correction algorithm is less efficient in these cases.

**Table 3** Average number of candidates for every encountered word length, and the uncorrected ratio

Word length	1	2	3	4	5	6	7	8	9	10	11	12	13
Candidates	26	345.7	316.4	115.1	97.3	36.3	18.9	4.6	3.2	2.1	1.7	1.9	1
Uncorrected Ratio %	72.7	28.7	34.7	22.7	16.9	9.9	4.6	3.8	1.9	3.5	6.6	7.2	14.3

The fact that shorter words are more difficult to correct is also shown in Figs. 6(a) and 6(b). There, we present the distribution of substitution errors by word length, firstly for all misspelled words, and then after applying the correction algorithm. Notice in Fig. 6(b) that after error correction, shorter words have a much higher contribution to the total number of errors. Indeed, words with length lower than 5 characters count for 69% of the uncorrected errors, but only represent 37% of the initial errors.



**Fig. 6** Error distribution for different word lengths. (a) Before error correction. (b) After error correction.

One way to decrease the high error rate among short words is to consider contextual information. For every word with less than 5 letters, we checked its neighbors and considered word bigrams in the correction algorithm. This can be seen as creating new words by concatenating adjacent ones. The length of the new word is the sum of the component words' length, and the prior probability is the bigram probability. This method reduces the substitution error rate from 1.25% to 1.06%, but has the drawback of increased number of computations and the need for a dictionary containing word bigrams and their frequencies.

Another possibility is to analyze part of speech  $n$ -grams, as proposed by Church in [2]. For example, if we encounter the one-letter word  $r$ , it will probably be an error. The most likely candidates are  $a$  (indefinite article) or  $I$  (pronoun). By checking

what part of speech is the following word, we can assume with high probability that in the case of a noun the word was  $a$ , and in the case of a verb it was  $I$ .

## 6 Conclusion

In this paper, we have presented an error correction algorithm designed for a five-key chording keyboard. For every typed word, it selects several possible candidates and then returns the most likely one. This is done using the MAP algorithm and the probabilities that one character is typed for another. These probabilities were determined experimentally. Even if the correction algorithm was designed for a specific keyboard and mapping, it can be easily adapted to other input devices by updating the confusion matrix.

The proposed error correction method reduces the substitution error rate from 7.60% to 1.25%, providing a considerable improvement compared to MsWord and iSpell (leading to substitution error rates of 3.12% and 3.94%, respectively). This advantage is due to the MAP algorithm which takes into account the prior distribution of words and the device-dependent confusion probabilities. We also showed that using personalized confusion matrices further reduces the substitution error rate from 1.25% to 1.15%. Using a larger dictionary allows to correct more errors, with the cost of increased search time and memory requirements. Checking word bigrams also decreases the error rate, but again, with the cost of increased complexity.

We have only focused on substitution errors because they represent more than 75% of the total errors. Improving the algorithm by also considering other error types such as missing or extra characters, or words which are not properly separated by white space characters or punctuation signs will further reduce the overall error rates. Another option for future work is to implement an adaptive approach, starting with a common confusion matrix and updating it based on what one types. Words which are typed more often can have their prior probability increased, becoming more likely than other candidates. One should also be able to add new words to the dictionary.

The comparison between our algorithm, MsWord and iSpell was done by only analyzing the first proposed candidate. We chose this approach because one possible use of the chording keyboards is in dynamical environments, like walking in crowded places or riding a bike, when users cannot continuously look at the typed text. Therefore, the error correction mechanism should run automatically, without requiring user supervision. In more static situations (for example when the keys are placed around a computer mouse, allowing typing and screen navigation with only one device), the most likely candidates can be displayed, and the user will choose the desired one.

## References

1. <http://www.kilgarriff.co.uk/bnc-readme.html>, November, 2013
2. Church, K.W.: A stochastic parts program and noun phrase parser for unrestricted text. In: Proceedings of the second conference on Applied natural language processing, ANLC '88, pp. 136–143. Association for Computational Linguistics, Stroudsburg, PA, USA (1988)
3. Goodman, J., Venolia, G., Steury, K., Parker, C.: Language modeling for soft keyboards. In: Proceedings of the 7th international conference on Intelligent user interfaces, IUI '02, pp. 194–195. ACM, New York, NY, USA (2002)
4. Kahan, S., Pavlidis, T., Baird, H.: On the recognition of printed characters of any font and size. Pattern Analysis and Machine Intelligence, IEEE Transactions on **PAMI-9**(2), 274–288 (1987)
5. Kay, S.: Fundamentals of statistical signal processing: estimation theory. Prentice-Hall (1993)
6. Kristensson, P.O., Zhai, S.: Relaxing stylus typing precision by geometric pattern matching. In: Proceedings of the 10th international conference on Intelligent user interfaces, IUI '05, pp. 151–158. ACM, New York, NY, USA (2005)
7. Kukich, K.: Techniques for automatically correcting words in text. ACM Comput. Surv. **24**, 377–439 (1992)
8. Lyons, K., Starner, T., Plaisted, D., Fusia, J., Lyons, A., Drew, A., Looney, E.W.: Twiddler typing: one-handed chording text entry for mobile phones. In: Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '04, pp. 671–678. ACM, Vienna, Austria (2004)
9. Noyes, J.: Chord keyboards. Applied Ergonomics **14**(1), 55 – 59 (1983)
10. Pollock, J.J., Zamora, A.: Automatic spelling correction in scientific and scholarly text. Commun. ACM **27**(4), 358–368 (1984)
11. Riseman, E., Hanson, A.: A Contextual Postprocessing System for Error Correction Using Binary n-Grams. IEEE Transactions on Computers **23**(5), 480–493 (1974)
12. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Parallel distributed processing: explorations in the microstructure of cognition, vol. 1. chap. Learning internal representations by error propagation, pp. 318–362. MIT Press, Cambridge, MA, USA (1986)
13. Sampson, G., Garside, R., Leech, G.: The Computational analysis of English : a corpus-based approach / edited by Roger Garside, Geoffrey Leech, Geoffrey Sampson. London ; New York : Longman (1987)
14. Sandnes, F., Huang, Y.P.: Non-intrusive error-correction of text input chords: a language model approach. In: Annual Meeting of the North American Fuzzy Information Processing Society, 2005. NAFIPS 2005, pp. 373 – 378 (2005)
15. H. Isaac and R.C. Nickerson and P. Tarasewich: Cell phone use in social settings : Preliminary results from a study in the United States and France. In: Decision Sciences Institute Conference (2004)
16. Tarniceriu, A., Dillenbourg, P., Rimoldi, B.: Single-handed typing with minimal eye commitment: A text-entry study. In: The Sixth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, UBICOMM '12. Barcelona, Spain (2012)
17. Tarniceriu, A., Dillenbourg, P., Rimoldi, B.: The effect of feedback on chord typing. In: The Seventh International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, UBICOMM '13. Porto, Portugal (2013)
18. Tarniceriu, A., Rimoldi, B., Dillenbourg, P.: Error correction mechanism for five-key chording keyboards. In: The 7th International Conference on Speech Technology and Human-Computer Dialogue, SpeD '13. Cluj-Napoca, Romania (2013)
19. Wagner, R.A.: Order-n correction for regular languages. Commun. ACM **17**(5), 265–268 (1974)
20. Yannakoudakis, E., Fawthrop, D.: The rules of spelling errors. Information Processing & Management **19**(2), 87–99 (1983)